International Journal of Computational Intelligence Systems and Applications. Volume 13, Issue 1, (2026) 26-34
http://www.cayley-nielson.com/ijcisa/

Hospital Ward Monitoring System Based on Arm-Linux

Xu Shuizhu[1], Shao Yongfeng[2], Wang Qianqian[3], Guo Chang[4]
[1,2,3,4]Xi'an Technological University, Xi'an, China
[1]Corresponding author
E-mail: [1]*3067738309@qq.com*, [2]*1218942993@qq.com,*[3]*169279356@qq.com,* [4]*2101527206@qq.com*

Abstract

This study adopts the Linux + Arm architecture to design a hospital ward monitoring system based on the mini2440 development board. Centered on the mini2440 as the core, the system is equipped with a DHT11 temperature and humidity sensor (5~95%RH, -20~+60℃) to realize real-time collection of ward temperature and humidity. It monitors the smoke concentration in the air via an MQ-2 smoke sensor (125~10000 ppm smoke, 300~10000 ppm combustible gas) and measures the light intensity through a BH1750 light intensity sensor (0~65535 lx). Meanwhile, a TCP data transmission module is utilized to guarantee real-time uploading of monitoring data.On the software side, driver programs are developed based on the Linux operating system to achieve sensor data reading and preprocessing, and multithreading technology is adopted to ensure concurrent processing of data collection and transmission. The data storage module designed at the software layer employs local file storage, with a storage efficiency of one record written per second, enabling the display and review of historical environmental data. The data collection module maintains a stable period of once per second. Medical staff and patients can view real-time ward environmental parameters through a 7-inch LCD screen, which refreshes every second to ensure smooth data presentation and real-time digital updates.

Keywords: Arm, Linux, Environmental Monitoring, Hospital, Ward Monitoring.

1. Introduction

The environmental quality of hospital wards directly affects patients' rehabilitation efficiency, the prevention and control of nosocomial cross-infection, and the work experience of medical staff. Accurate monitoring and real-time regulation of parameters such as temperature and humidity, air quality, and light intensity are core requirements for modern hospital management[1-2]. At present, the environmental monitoring of traditional hospital wards mostly adopts the manual single-point detection mode, which suffers from problems such as low monitoring efficiency, strong data lag, and failure to realize real-time early warning. Some existing intelligent monitoring systems also have shortcomings including high hardware costs, single monitoring parameters, and poor software compatibility, making it difficult to promote them on a large scale in primary hospitals[3-4]. With the rapid development of embedded technology, sensor technology and Internet of Things technology, technical support has been provided for the intelligent and integrated transformation of hospital ward environmental monitoring[5]. The Linux operating system, featuring open-source nature, stability and abundant driver support, has become a mainstream development scheme for embedded monitoring systems when combined with the Arm architecture[6]. In response to the actual monitoring needs of hospital wards, this study designs a multi-parameter intelligent monitoring system based on Arm-Linux, which realizes real-time collection, transmission, storage and visual display of temperature and humidity, smoke concentration and light

intensity. With high cost performance and strong scalability, the system can effectively make up for the deficiencies of traditional monitoring methods and adapt to the ward environmental management needs of hospitals at all levels.

2. Research Content

2.1. Overall System Design

This system leverages embedded Internet of Things technology to embed various sensors into monitoring objects for different hospital ward environments. It integrates the hospital environmental domain through computers and cloud platforms, enabling collaboration between human society and environmental business systems. The Arm-Linux-based hospital ward monitoring system is an instrument used to detect various parameters in hospital ward environments. The working process of this instrument can be divided into three steps: data collection, data processing, and data display. The corresponding hardware implementations are three subsystems: the sensor group, the S3C2440 main control board, and the LCD display module. The overall system design process is shown in Figure 1.

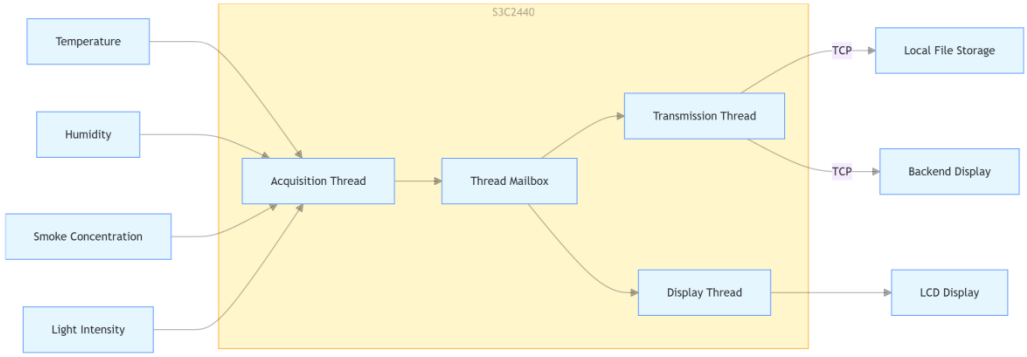

Fig. 1. Overall design flowchart

2.2. System Hardware Acquisition Module

The data acquisition module mainly uses a variety of sensors to collect various environmental conditions in hospital wards. The mini2440 main control board[7] serves as the core of the system for data processing, whose specific functions include data storage, data calibration and filtering, real-time analysis, etc. The acquisition module adopts the DHT11 temperature and humidity sensor[8], MQ-2 smoke sensor[9] and BH1750[10] light intensity sensor to monitor the hospital environment. Based on an 800*480 LCD screen, the data display module presents various monitoring parameters in real time, which is intuitive and easy to understand. On this basis, a historical record function is added to display the trend of historical data at a glance, helping medical staff make better judgments and facilitating system maintenance.

2.3. System Software Design

2.3.1. Overall Framework Of System Software

The system software is developed based on the Linux 2.6.32.2 operating system, adopting a modular and multi-threaded design concept. It is divided into a driver layer and an application layer as a whole. The driver layer is responsible for the driver development of sensors and hardware devices, while the application layer is in charge of data processing, transmission, storage and display. Each module operates

independently and realizes data interaction through thread mailboxes, ensuring the stability, efficiency and scalability of the software.

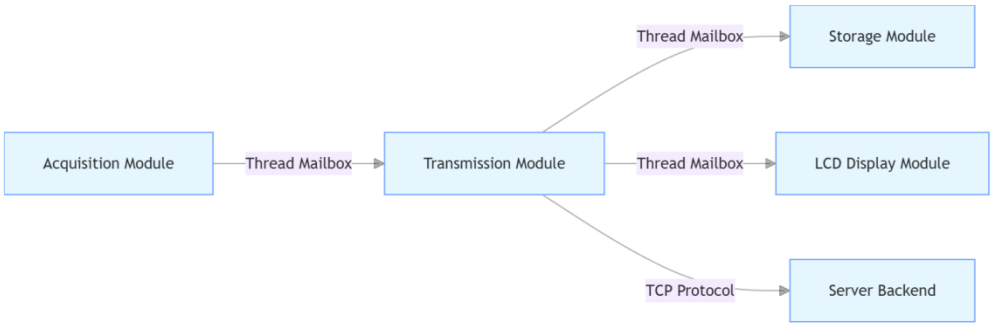

Fig. 2. Software framework flowchart

2.3.2. Development Environment Setup

Embedded system development adopts the mode of "cross-compilation on a PC + operation on a development board". First, the development environment is set up in the Ubuntu system, and the u-boot bootloader is flashed to complete hardware initialization and kernel booting. The TFTP and NFS transmission protocols are configured to realize file sharing and fast transmission between the PC and the development board[11]. The arm-Linux-gcc cross-compilation toolchain is installed, and environment variables are configured to compile programs for the ARM architecture. The Linux 2.6.32.2 kernel is ported to the mini2440 development board, and the root file system is configured to build a basic platform for software operation.

2.3.3. Sensor Driver Design

Under the Linux system, drivers are mainly divided into three categories: character device drivers, block device drivers, and network device drivers[12]. The DHT11 temperature sensor uses GPIO communication, and the macro instructions of the S3C2440 GPIO subsystem are employed to pull the signal level high and low, accurately matching the sensor's timing requirements. The MQ-2 sensor adopts ADC communication, using an interrupt and wait queue mechanism; the interrupt triggers the sampling timing, and the wait queue resolves the time delay issue in data conversion. The BH1750 sensor uses I2C communication. Based on the Linux I2C subsystem, the driver is abstracted into two parts: hardware-related device and logic-related driver, realizing driver layering and separation and improving code reusability[13]. After compilation, the driver generates a .ko file, which is loaded into the development board kernel via the insmod command to enable communication between the sensors and the core processing module.

2.3.4. Data Transmission Program Design

In this system, thread mailboxes are used for data transmission in lower computer communication. Thread mailboxes implement asynchronous communication by passing messages (Mails) between threads[14]. Each message contains specific data or instructions, and threads can send or receive messages through mailboxes to coordinate task execution. Essentially, a thread mailbox is a fixed-size queue (First-

In-First-Out, FIFO) for storing pending messages. It features asynchronous communication, low coupling, ease of use, and low resource overhead.

The mini2440 is connected to the upper PC via a network cable and a serial port, so the communication methods between them mainly rely on network communication and serial cables. However, compared with serial cables, network transmission is more stable and has lower requirements for hardware connections. In this system, it is mainly implemented using socket in network communication.

Table 1. Comparison between TCP and UDT

	TCP	UDP
Connection	Connection-oriented	Connectionless
reliability	Secure and reliable	Unsecure and unreliable
resource overhead	High resource overhead	Low resource overhead
Layert	Transport layer	Transport layer
Data Format	Byte stream	Datagram

2.3.5. Display Program Design

The mini2440 is equipped with an 800*480 LCD display screen, connected via an RGB cable. The LCD can operate properly and display collected information in real time through three timing signals: VSYNC, HSYNC, and VCLK. Develop a socket communication program based on the TCP protocol to send collected real-time data to the host computer server, realizing remote background monitoring. The TCP protocol ensures orderly and lossless data transmission through mechanisms such as three-way handshake, four-way wave, and acknowledgment response[15].

The Linux system provides a UI design model called Frame Buffer. Frame Buffer (frame buffer) is a memory area for storing image data and serves as a key data interface between the graphics card and the display[16]. It directly maps memory addresses to pixels on the screen, with each memory location corresponding to a specific position on the display. By writing RGB values to these specific locations and filling all points, a complete UI interface can be formed. Compared with other complex UI designs, applications can directly write to Frame Buffer to update screen content without relying on complex graphics APIs, making operations more convenient and user-friendly. However, compared with other UI schemes, Frame Buffer is relatively inconvenient for displaying characters such as Chinese characters. To display a character, it requires a corresponding font module to write values to the matching RGB points, and the writing positions must be manually controlled; otherwise, the designed UI interface will appear disorganized.

For displaying characters such as Chinese characters, two methods are commonly used: one is to extract font modules independently, and the other is to directly use a font library. The latter offers greater flexibility and convenience compared with the former.

Under the Linux operating system, Frame Buffer is a character device with a major device number of 29, so it can be operated using standard character device methods. The general process for graphical display

using the Frame Buffer device is as follows: open the /dev/fb0 device file to obtain device access permissions; use the ioctl system call to query the device's fixed parameters (including video memory size, line stride, etc.) and variable parameters (such as pixel depth, color format, resolution, etc.); map the video memory space to user space via a function, allowing direct reading and writing of video memory to control screen display; After completing image drawing or content display, the munmap function must be called to unmap the memory mapping; otherwise, it will cause problems in the program and damage the hardware. finally, close the device file to release resources. Figure 3 shows the Frame Buffer operation flow chart.

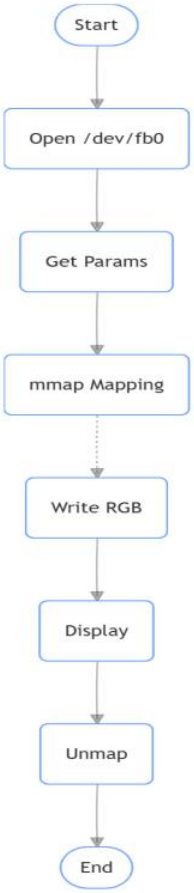

Fig. 3. Frame Buffer operation flowchart

3. System Testing and Result Analysis

3.1. Test Environment and Scheme

To verify the functional effectiveness and operational stability of the system, a simulated hospital ward test environment was built, and standardized hardware connections were completed: sensors, LCD screens were connected to the mini2440 development board, the development board was connected to the host computer (PC with Ubuntu 18.04 system) via a network cable, and the development board was powered on to start the system. The test adopted the method of separate module testing plus system joint testing. First,

functional tests were conducted on the data acquisition, display, transmission, and storage modules respectively to verify whether each module met the design requirements; then a 72-hour uninterrupted joint test was carried out to simulate the actual operating scenarios of the ward, and environmental parameters were randomly changed to verify the overall operational stability of the system.

3.2. Data Acquisition Module

After connecting the wires of the development board correctly, enter the command sudo minicom in the Linux terminal. The -con option can be added to enable color changes in the development board terminal. After entering the command, you will enter a human-computer interaction interface. At this point, entering the command tftp uImage 0x30008000 will download the Linux kernel to the register address 0x30008000. Then enter the command bootm to access the root file system in the development board terminal. Before running the program, you need to use the insmod command to load the driver files of each sensor; otherwise, the program will not recognize the sensors and thus fail to run. However, testing all functions simultaneously may cause damage to the sensor hardware, leading to problems in the entire system. Therefore, each sensor needs to be tested carefully. Figure 4 shows the test results of the DHT11 temperature and humidity sensor, Figure 5 shows the test results of the BH1750 light intensity sensor, and Figure 6 shows the test results of the MQ-2 smoke sensor.

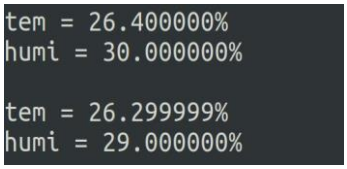

Fig. 4. Test results of temperature and humidity sensors

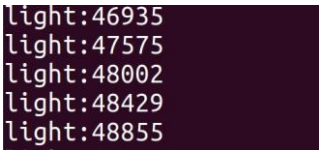

Fig. 5. Test results of light intensity sensor

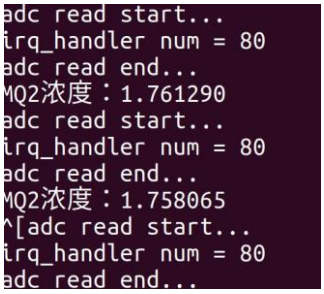

Fig. 6. Smoke sensor test results

3.3. Data Display Module

After the entire program runs, the FrameBuffer technology will be used to map the graphics card memory, and different RGB values will be written to the corresponding pixel points to form images, Chinese characters and other character symbols. Figure 7 shows the display results on the screen and at the bottom of the terminal after the acquisition module sends the collected data to the display module when the system is running. Meanwhile, the collected data will also be printed at the bottom of the terminal to prevent errors during data transmission. Figure 8 shows the display of the collected results at the bottom of the terminal.

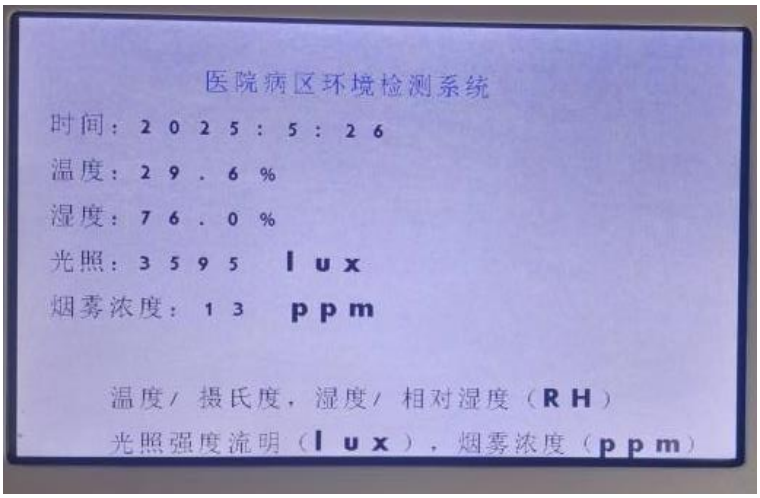

Fig. 7. Data acquisition screen display result chart

```
tem = 29.6%
humi = 76.0%
adc read start...
irq_handler num = 80
adc read end...
smoke:13
light:3595
```

Fig. 8. Display results of the data acquisition terminal

3.4. Testing of the Transmission Module and Storage Module

Before powering on the development board, the server program must be started first; otherwise, the collected information cannot be sent to the server. After both ends are started, the client will transmit the collected information to the server via the TCP protocol, and the server will print the received information on the server terminal for convenient background monitoring. Figure 9 shows the server running results. Meanwhile, the client will create a .txt file to store the collected information in the local memory of the development board. Figure 10 shows the result of printing the contents from the stored file.

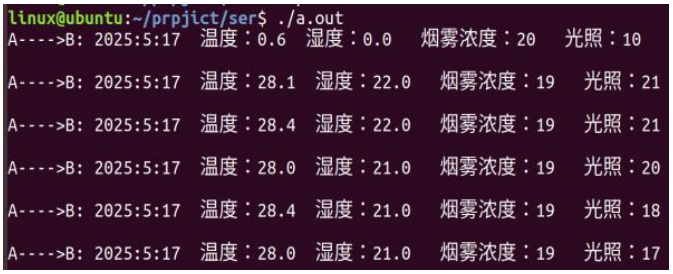

Fig. 9. Server-side received information result

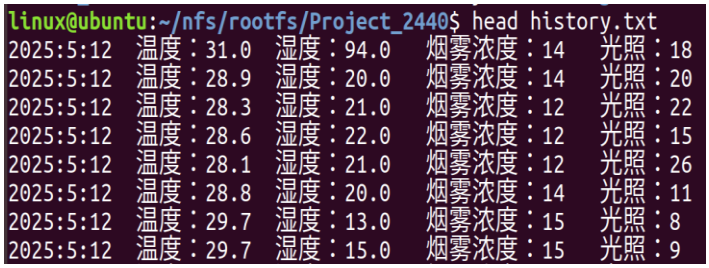

Fig. 10. Historical information.

4. Conclusion

The multi-parameter intelligent monitoring system for hospital wards designed in this study, which is based on Arm-Linux, integrates embedded technology, sensor technology, network communication technology and Linux programming technology. It realizes the real-time collection, transmission, storage and visual display of ward temperature and humidity, smoke concentration and light intensity. The system features reasonable hardware selection and low cost, as well as modular software design and efficient operation. Tests have verified that the system achieves accurate collection, stable transmission and intuitive display, and is characterized by low power consumption, easy operation and good scalability. It can effectively solve the pain points of traditional hospital ward environmental monitoring, provide a implementable technical solution for the refined and intelligent management of the hospital environment, and has high practical application value and promotion prospects.

Acknowledgment

This work was supported by the Training Program of Innovation and Entrepreneurship for Undergraduates. Grant No.S202410702111 and S202410702104. The authors would like to express their sincere gratitude to the funders for their generous financial support. Special thanks also go to all members of the research team for their assistance in data collection and analysis.

References

[1] Lin K, Chen K, Wang J B, et al. Time series study on environmental hygiene and nosocomial infection based on generalized additive model (GAM)[J]. Chinese Journal of Infection Control, 2024, 23(7): 798-805.

[2] Hui Y Y, Fu G J, Zhao Y. Application of information system in closed-loop management of hospital environmental hygiene monitoring[J]. Chinese Journal of Disinfection, 2022, 39(12): 902-904, 907.

[3] Chen L F. Research and implementation of hospital temperature and humidity management system based on hybrid network[D]. Zhejiang: Zhejiang University, 2023.
[4] Shoubridge A P, Brass A, Elms L, et al. Atmospheric CO_2 monitoring to identify zones of increased airborne pathogen transmission risk in hospital settings[J]. Current Diabetes Reports, 2024.
[5] Liang Y. Practical analysis of internet of things technology in the era of big data[J]. Computer & Information, 2025, 37(7): 32-34.
[6] Tang J. Practical exploration of artificial intelligence enabling the teaching of Linux Operating System Application[J]. China Informatization, 2025(4): 72-73.
[7] Du M Y. Development of home remote monitoring system based on mini2440[J]. Information Technology and Informatization, 2018(10): 52-54, 57.
[8] Chen J X. Application of DHT11 digital temperature and humidity sensor in greenhouse control system[J]. Shandong Industrial Technology, 2016(18): 120.
[9] Xie Y C, Yang L, Yan J. Design of smoke detection alarm based on MQ-2 sensor[J]. Computer Measurement & Control, 2021, 29(8): 255-259.
[10] Liu Q, Sun K, Yang J Q, et al. Design and application evaluation of chronic wound surface humidity detector[J]. China Medical Equipment, 2023, 20(7): 157-161.
[11] Chen L C, Pu Z K, Shen Y. Design of motion controller based on TFTP protocol[J]. Electronic Design Engineering, 2024, 32(17): 78-82.
[12] Wen J L. Research on runtime security of Linux based on kernel driver[D]. Changsha: Hunan University, 2023.
[13] Li Y, Wu H, Liu Z K. Design of image acquisition and display system based on Linux kernel framebuffer[J]. Computer Measurement & Control, 2018, 26(3): 124-128.
[14] Su W Y. Research and implementation of mobile agent communication technology in mobile environment[D]. Changsha: National University of Defense Technology, 2002.
[15] Cao Y H, Yan G Q. Application of Modbus-TCP to Modbus-RTU gateway in industrial automation systems[J]. Paper Making Equipment and Materials, 2025, 54(4): 25-27.
[16] Liang Z, Guo H T, Wu Y Q. Linux framebuffer for image display[J]. Electronics World, 2018(19): 81, 83.